Mining Plausible Hypotheses from the Literature via Meta-Analysis

Jooyong Yi
Ulsan National Institute of Science and Technology
South Korea
jooyongyi@acm.org

Vladimir Ivanov Giancarlo Succi
Innopolis University
Russia
{v.ivanov, g.succi}@innopolis.ru

***Abstract*—Meta-analysis is highly advocated in many fields of empirical research such as medicine and psychology, due to its capability to synthesize quantitative evidence of effects from the literature, based on statistical analysis. However, the adoption of meta-analysis to software engineering is still suffering from inertia, despite the fact that many software engineering researchers have long been arguing the need for it. As an attempt to move beyond the lockstep, we in this paper explore a different use of meta-analysis. Our proposition is that meta-analysis is useful for mining hypotheses because their plausibility is backed by evidence accumulated in the literature, and thus researchers could focus their effort on the areas that are of particular need. We assess our proposition by conducting a lightweight case study on the literature of defect prediction. We found that three out of five hypotheses we extract from our meta-analysis were indeed investigated in separate papers, indicating the usefulness of our approach. We also recognize two uninvestigated hypotheses whose validity we plan to investigate in the future.**

I. Introduction

In this paper, we provide a new perspective on an existing idea we believe worth of much more attention. The existing idea we advocate here is meta-analysis. Meta-analysis refers to the statistical analysis and synthesis of results from a series of primary studies, i.e., individual studies under investigation [3], [10]. Common use of meta-analysis is to synthesize consolidated evidence from a body of primary studies often containing conflicting results.

Meta-analysis has been advocated and widely used in many fields of empirical research, such as medicine [10], psychology [1], education [5], and business [13], enabling evidence-based practice and decision-making. In software engineering, despite the community-wide attention to Empirical and Evidence-Based Software Engineering movement initiated by Victor Basili, Barry Boehm, and Dieter Rombach in the 80's [2], the generalization of findings across experiment in general, and the adoption of meta-analysis specifically remain limited to a fraction of papers such as [22], [25].

Further adoption of meta-analysis to software engineering research has been called for by many researchers, for example in [6], [14], [20], [24], to name a few. Our work is in line with those pieces of work, but also moves beyond them, by advocating the new use of meta-analysis for *mining hypotheses whose plausibility is backed by evidence accumulated in the literature*. Considering that setting up a proper hypothesis is a critical first step of empirical research, and performing a hypothesis test typically takes substantial amount of time of researchers, it is our proposition that using meta-analysis to mine a hypothesis would produce a significant advance of the discipline, since researchers could focus their effort on the areas that are of particular need.

How can hypotheses be mined from the literature via meta-analysis? As will be shown with a concrete example in this paper, we make use of two standard mechanisms of meta-analysis: a forest plot and subgroup analysis (an example is shown in Fig. 1). With a forest plot, one can see whether or not similar results are observed across multiple studies in a consistent manner. If that is the case, the observed consistency can form a plausible hypothesis. Otherwise, subgroup analysis can be performed to see which factors contribute to inconsistent observations, and a hypothesis can be made accordingly for each of those factors.

In other fields such as medicine, meta-analysis has already been used in suggesting a new hypothesis [7]. Our key contribution in this paper is to provide early assessment of using meta-analysis in generating evidence-backed hypotheses in the context of software engineering. We assess our proposition by (1) extracting five hypotheses from the literature on defect prediction via meta-analysis, and (2) subsequently checking whether those hypotheses were indeed tested in separate papers in the literature. The rationale of the second step is as follows. The fact that a hypothesis was tested in the literature indicates its worthiness for consideration. We found that two hypotheses were indeed studied in recent papers, one hypothesis was partially investigated, and two hypotheses remain to be studied.

II. Background

A common way to perform a meta-analysis is to construct a forest plot such as Fig. 1. Information that can be shown with a forest plot includes the following.

- Individual Studies: In Fig. 1, each square represents the mean effect size of the corresponding study, with its size proportional to its weight reflecting the precision of the study. Meanwhile, each horizontal line crossing across a square represents the 95% confidence interval of the corresponding study.
- Summary Effect: In Fig. 1, the diamond in the bottom represents the summary effect, that is, the weighted mean of

the individual effects. The center of the diamond represents the summary effect size, and the width of the diamond represents the 95% confidence interval.

- Heterogeneity: A forest plot also shows how heterogeneous individual studies are to each other quantitatively. I^2 is the ratio of true heterogeneity in observed effects to total variation. τ^2 is the variance of true effect sizes, and reflects the amount of true heterogeneity. A p-value shows the statistical significance for the null hypothesis that $\tau^2 = 0$. In general, high values of I^2 and τ^2 imply high heterogeneity.
- Subgroups: In Fig. 1, individual studies are divided into two subgroups, and their summary effects and heterogeneity are shown in the bottom of each subgroup.

III. A Case Study

To assess our proposition, we conducted a lightweight case study on defect prediction.

A. Paper Identification and Selection

To search for papers for our meta-analysis, we consulted ACM Digital Library and IEEE Xplore. As for the key phrase, we started with "defect prediction" or "fault prediction", and increased the pool using a snowballing method. Out of 956 papers elicited in the initial search, we selected the 13 papers listed in Table I satisfying the following criteria. In some papers such as Sun-TSMC12 [27] and Ozturk-ASCJ16 [30], two separate experimental results conducted in two different settings are available, and we distinguish them by putting suffixes _1 and _2 respectively.

1) The paper should be about defect prediction. In this paper, we only consider within-project defect prediction.
2) The paper should contain experimental results about the performance of defect prediction. In this paper, we only consider AUC (Area Under the Curve).
3) The experiments conducted in the paper should involve multiple classifiers. In this study, we only consider the following three techniques: C4.5, NB (Naive Bayes) and RF (Random Forest), which appear most frequently in our primary studies.

B. Results

Fig. 1–3 show the forest plots for the mean difference of AUC between three classifiers, C4.5, NB and RF. We compute the mean difference of AUC by comparing the AUC values of classifiers in each dataset available in a primary study.[1] The 'Total' column of each forest plot shows the number of datasets available in respective primary studies. Each plot has two subgroups, and how we divided subgroups will be explained shortly. We used the meta function of the meta package of R to construct forest plots.[2]

Overall Observations. Across all three plots, the summary effect (represented with the diamond in the bottom line of each

[1]While acknowledging that this is not the best approach (an AUC value for each dataset typically represents a mean value from multiple experiments), dispersion is usually not reported in the literature on defect prediction.

[2]https://cran.r-project.org/web/packages/meta/index.html

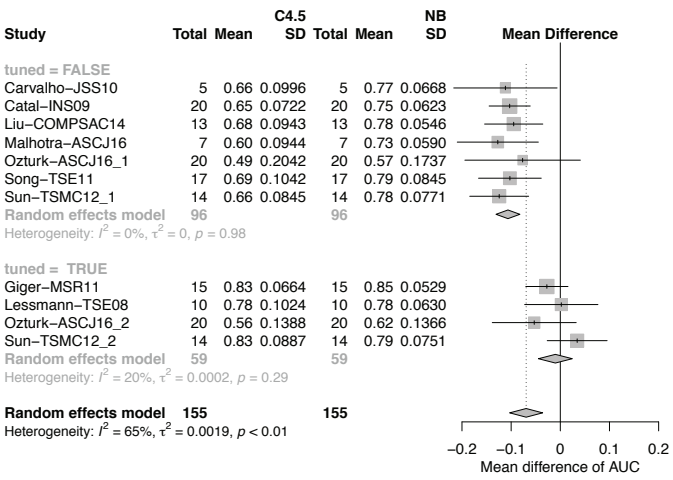

Fig. 1. A forest plot for the mean difference of AUC between C4.5 and NB

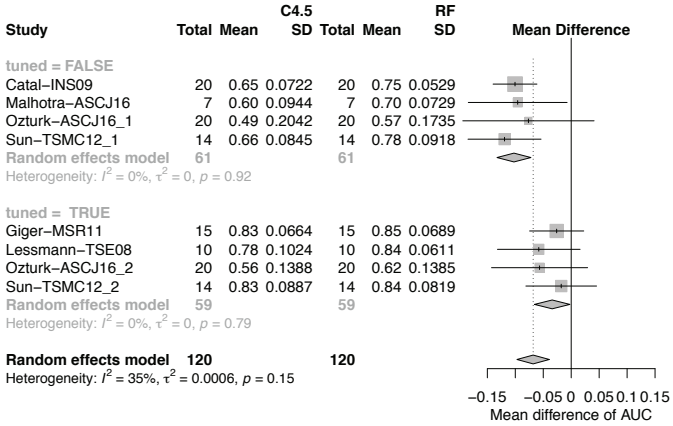

Fig. 2. A forest plot for the mean difference of AUC between C4.5 and RF

plot) in common does not overlap with the zero point. This implies that the null hypothesis – there is no performance difference between two classifiers C4.5 and NB (likewise between C4.5 and RF and NB and RF) – is rejected with statistical significance. This result is in contrast with an earlier study conducted by Lessmann et al [16], which leads to our first hypothesis:

> H0) Performance differences between classifiers can vary with statistical significance, depending on confounding factors.

Finding Potential Confounding Factors. The most interesting keyword from H0 would be "confounding factors." What would be confounding factors and how do they affect the performance differences between classifiers? These are research questions many researchers asked in the past, and also will ask in the future.

How does meta-analysis help in finding confounding factors? For this purpose, we conduct simple subgroup analyses. We divide the studies into Lessmann-TSE08 vs. non-Lessmann-TSE08, where Lessmann-TSE08 refers to the study of Lessmann et al [16]. As mentioned, no significant performance difference between classifiers is exhibited in Lessmann-TSE08. Initially, the Lessmann-TSE08 subgroup only contains Lessmann-TSE08, and we repeatedly move a primary study in

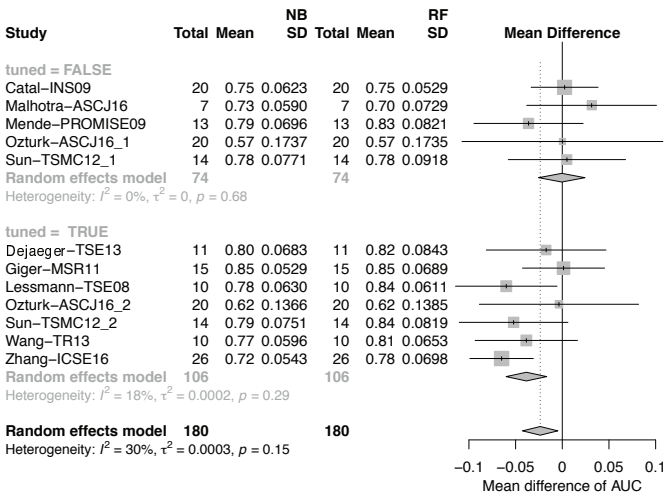

Fig. 3. A forest plot for the mean difference of AUC between NB and RF

the non-Lessmann-TSE08 subgroup to the Lessmann-TSE08 subgroup in the following way. At each iteration, we identify a primary study in the non-Lessmann-TSE08 subgroup whose effect size is closest to Lessmann-TSE08. Then, we check if there is a factor that distinguishes the study from the remaining ones in the non-Lessmann-TSE08 subgroup. If we find a distinguishing factor, we move the study to the Lessmann-TSE08 subgroup. Otherwise, we keep the study in the non-Lessmann-TSE08 subgroup, and consider the next study.

Table I shows the potential confounding factors we obtained as a result. Each column of the table represents a factor that distinguishes the non-Lessmann-TSE08 subgroup (upper) from the Lessmann-TSE08 subgroup (lower). 'Param' denotes the use of parameter tuning of classification models, 'Multi-class' the use of multi-classification techniques, 'Cleaned' the use of data cleaning, 'Norm' the use of metrics normalization, and 'Sem' the use of semantics-capturing metrics such as fine-grained source code changes [12].

Each of the three forest plots (Fig. 1–3) shows the summary effects and heterogeneity of the two subgroups in grey texts, where the upper part and the lower part respectively represents the non-Lessmann-TSE08 subgroup and Lessmann-TSE08 subgroup. Overall, it is commonly observed across the three forest plots that the upper subgroup shows little heterogeneity, as shown with the zero I^2 values (implying that there is little true heterogeneity other than random error), the zero τ^2 values, and the non-significant p-values (thus, the null-hypothesis that $\tau^2 = 0$ is not rejected). Meanwhile, it is also commonly observed across the three forest plots that the overall heterogeneity is larger than in the upper subgroup. Meta-analysis provides evidence to hypothesize that the factors we identified may contribute to the observed heterogeneity.

Extracted Hypotheses. Based on the five potential confounding factors we identified, we make the following five hypotheses accordingly. When making hypotheses, we consider the fact that the identified confounding factors were used to improve the performance of defect prediction. *Performance of defect prediction improves to a different degree depending on a classification model by use of H1) parameter tuning of classification models, H2) multi-classification techniques, H3) data cleaning, H4) metrics normalization, or H5) semantics-capturing metrics.*

TABLE I
POTENTIAL CONFOUNDING FACTORS THAT DISTINGUISH THE NON-LESSMANN-TSE08 SUBGROUP (UPPER) FROM THE LESSMANN-TSE08 SUBGROUP (LOWER)

Study	Param	Multiclass	Cleaned	Norm	Sem
Carvalho-JSS10 [8]	N	N	N	N	N
Catal-INS09 [4]	N	N	N	N	N
Liu-COMPSAC14 [17]	N	N	N	N	N
Malhotra-ASCJ16 [18]	N	N	N	N	N
Ozturk-ASCJ16_1 [30]	N	N	N	N	N
Song-TSE11 [26]	N	N	N	N	N
Sun-TSMC12_1 [27]	N	N	N	N	N
Mende-PROMISE09 [19]	N	N	N	N	N
Lessmann-TSE08 [16]	**Y**	N	N	N	N
Dejaeger-TSE13 [9]	**Y**	N	N	N	N
Wang-TR13 [29]	**Y**	N	N	N	N
Sun-TSMC12_2 [27]	N	**Y**	N	N	N
Ozturk-ASCJ16_2 [30]	N	N	**Y**	N	N
Zhang-ICSE16 [31]	N	N	N	**Y**	N
Giger-MSR11 [12]	N	N	N	N	**Y**

Tested Hypotheses in the Literature. In this study, our goal is to assess the usefulness of meta-analysis in generating hypotheses. Thus, one obvious possibility is to test each of the extracted hypotheses in a separate dedicated study. In this paper, we instead take the following lightweight approach as a necessary precursor. We surveyed the literature on defect prediction to see whether there exist studies that investigated the five hypotheses, H1–H5. The existence of such papers indirectly indicate the usefulness of meta-analysis in generating hypotheses. We do not consider H0, since it is a generalized hypothesis of H1–H5.

Hypothesis H1 was thoroughly investigated by Tantithamthavorn et al [28]. They optimized the parameters of classification models using an off-the-shelf automated parameter optimization technique, and investigated how much performance improvement is made in each classification technique. According to their study, performance of defect prediction indeed improves differently in each classification technique. Hypothesis H3 is related to the study of Ghotra et al [11] that reported that after cleaning data, performance differences between different classification techniques become more prominent than before cleaning data. Regarding H4, Nam et al [21] showed that the performance of logistic regression improves when z-score normalization [15] is applied to metrics. However, investigation into how z-score normalization affects the performance of different classification techniques was not their objective. For the remaining two hypotheses, H2 and H5, we could not find relevant studies in the literature. We plan to conduct experiments to validate these remaining hypotheses.

IV. DISCUSSION

Novelty. It is customary in empirical research that researchers conjecture hypotheses based on the existing work, be it done based on the intuition of a researcher or done more

explicitly through a systematic literature review. The novelty of our approach is in making this hypothesis-mining process rigorous and systematic by exploiting statistical information provided through meta-analysis, which can provide researchers with high confidence and motivation to set out to test the conjectured hypothesis.

Automation. While in this paper, we identify confounding factors leading to hypotheses in a manual way, the same process can be performed automatically if raw data from the existing studies (experimental results and potential confounding factors) are already available. More specifically, subgroup analysis can be performed with various confounding factors and their combinations, while comparing statistical information such as I^2, τ^2 and p-value obtained at each case with certain thresholds (e.g., the zero I^2 value). This paper advocates the possibility of such a more automated and systematic way of mining hypotheses through meta-analysis.

Call for Collaboration. The main bottleneck in using our approach is the manual effort in collecting raw data from papers. To alleviate this manual effort, we advocate community-wide collaboration to share raw data. Just like sharing datasets has catalyzed research in software engineering (with the PROMISE dataset [23] that sparked the proliferation of defect prediction research being the primary example), we believe that sharing raw data of experiments — including experimental results and potential confounding factors — has a good potential for another catalysis. We share our data in https://github.com/jyi/MHMA.

V. Related Work and Conclusion

Shepperd et al [25] showed through meta-analysis that the performance of defect prediction is affected more strongly by who conducted experiments than which classification technique is used. Rafique el al [22] conducted a meta-analysis to investigate the impact of Test-Driven Development (TDD) on code quality and productivity.

In this study, we explore the possibility of using meta-analysis from the opposite angle. Instead of deriving a conclusion, we use meta-analysis to mine hypotheses. We envision that setting up a hypothesis based on meta-analysis would benefit the research community, not only because it helps researchers find plausible hypotheses, but also because it can provide evidence why the hypothesis matters.

References

[1] M. R. Barrick and M. K. Mount. The big five personality dimensions and job performance: a meta-analysis. *Personnel psychology*, 44(1):1–26, 1991.

[2] B. Boehm, H. D. Rombach, and M. V. Zelkowitz. *Foundations of empirical software engineering: the legacy of Victor R. Basili*. Springer Science & Business Media, 2005.

[3] M. Borenstein, L. V. Hedges, J. P. Higgins, and H. R. Rothstein. *Introduction to meta-analysis*. John Wiley & Sons, 2011.

[4] C. Catal and B. Diri. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences*, 179(8):1040–1058, 2009.

[5] P. A. Cohen, J. A. Kulik, and C.-L. C. Kulik. Educational outcomes of tutoring: A meta-analysis of findings. *American educational research journal*, 19(2):237–248, 1982.

[6] D. S. Cruzes and T. Dybå. Research synthesis in software engineering: A tertiary study. *Information and Software Technology*, 53(5):440–455, 2011.

[7] M. Cucherat, J.-P. Boissel, A. Leizorovicz, and M. C. Haugh. EasyMA: a program for the meta-analysis of clinical trials. *Computer methods and programs in biomedicine*, 53(3):187–190, 1997.

[8] A. B. De Carvalho, A. Pozo, and S. R. Vergilio. A symbolic fault-prediction model based on multiobjective particle swarm optimization. *Journal of Systems and Software*, 83(5):868–882, 2010.

[9] K. Dejaeger, T. Verbraken, and B. Baesens. Toward comprehensible software fault prediction models using bayesian network classifiers. *TSE*, 39(2):237–257, 2013.

[10] R. DerSimonian and N. Laird. Meta-analysis in clinical trials. *Controlled clinical trials*, 7(3):177–188, 1986.

[11] B. Ghotra, S. McIntosh, and A. E. Hassan. Revisiting the impact of classification techniques on the performance of defect prediction models. In *ICSE*, pages 789–800, 2015.

[12] E. Giger, M. Pinzger, and H. C. Gall. Comparing fine-grained source code changes and code churn for bug prediction. In *MSR*, pages 83–92. ACM, 2011.

[13] J. K. Harter, F. L. Schmidt, and T. L. Hayes. Business-unit-level relationship between employee satisfaction, employee engagement, and business outcomes: a meta-analysis. *Journal of applied psychology*, 87(2):268, 2002.

[14] M. Jørgensen, T. Dybå, K. Liestøl, and D. I. Sjøberg. Incorrect results in software engineering experiments: How to improve research practices. *Journal of Systems and Software*, 116:133–145, 2016.

[15] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[16] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *TSE*, 34(4):485–496, 2008.

[17] S. Liu, X. Chen, W. Liu, J. Chen, Q. Gu, and D. Chen. Fecar: A feature selection framework for software defect prediction. In *COMPSAC*, pages 426–435. IEEE, 2014.

[18] R. Malhotra. An empirical framework for defect prediction using machine learning techniques with android software. *Applied Soft Computing*, 49:1034–1050, 2016.

[19] T. Mende and R. Koschke. Revisiting the evaluation of defect prediction models. In *PROMISE*, page 7. ACM, 2009.

[20] J. Miller. Applying meta-analytical procedures to software engineering experiments. *Journal of Systems and Software*, 54(1):29–39, 2000.

[21] J. Nam, S. J. Pan, and S. Kim. Transfer defect learning. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 382–391. IEEE, 2013.

[22] Y. Rafique and V. B. Mišić. The effects of test-driven development on external quality and productivity: A meta-analysis. *TSE*, 39(6):835–856, 2013.

[23] J. Sayyad Shirabad and T. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

[24] M. Shepperd. Replication studies considered harmful. In *ICSE-NIER*, pages 73–76. ACM, 2018.

[25] M. J. Shepperd, D. Bowes, and T. Hall. Researcher bias: The use of machine learning in software defect prediction. *TSE*, 40(6):603–616, 2014.

[26] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu. A general software defect-proneness prediction framework. *TSE*, 37(3):356–370, 2011.

[27] Z. Sun, Q. Song, and X. Zhu. Using coding-based ensemble learning to improve software defect prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1806–1817, 2012.

[28] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto. Automated parameter optimization of classification techniques for defect prediction models. In *ICSE*, pages 321–332. IEEE, 2016.

[29] S. Wang and X. Yao. Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2):434–443, 2013.

[30] A. Zengin et al. How repeated data points affect bug prediction performance. *Applied Soft Computing*, 49(C):1051–1061, 2016.

[31] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan. Cross-project defect prediction using a connectivity-based unsupervised classifier. In *ICSE*, pages 309–320. ACM, 2016.